(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
21 mai 2004 (21.05.2004)

PCT

(10) Numéro de publication internationale
WO 2004/042572 A2

(51) Classification internationale des brevets[7] : G06F 9/46

(21) Numéro de la demande internationale :
PCT/FR2003/003257

(22) Date de dépôt international :
31 octobre 2003 (31.10.2003)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
02/13750 4 novembre 2002 (04.11.2002) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **OBERTHUR CARD SYSTEMS SA** [FR/FR]; 102, boulevard Malesherbes, F-75017 Paris (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **FLATTIN, David** [FR/FR]; 46, rue Jean-Jacques Rousseau, F-92150 Suresnes (FR). **LOUIS, Christophe** [FR/FR]; 12, allée de Persepolis, F-91400 Orsay (FR). **CONTRERAS, Javier** [FR/FR]; 8, avenue du Maine, F-75015 Paris (FR).

(74) Mandataire : **SANTARELLI**; 14, avenue de la Grande-Armée, Boîte postale 237, F-75822 Paris Cedex 17 (FR).

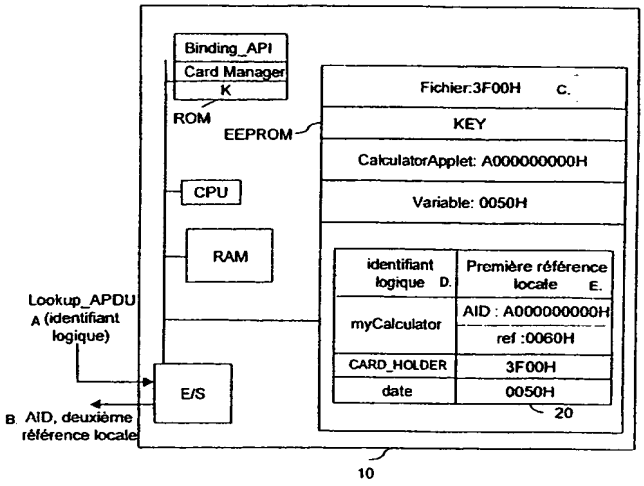
(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) États désignés *(régional)* : brevet ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Suite sur la page suivante]

(54) Title: MICROCIRCUIT CARD COMPRISING MEANS FOR PUBLISHING ITS COMPUTER OBJECTS

(54) Titre : CARTE A MICROCIRCUIT COMPORTANT DES MOYENS DE PUBLICATION DE SES OBJETS INFORMATIQUES



A...LOGIC IDENTIFIER
B...SECOND LOCAL REFERANCE
C...FILE 3 F00H
D...LOGIC IDENTIFIER
E...FIRST LOCAL REFERENCE

(57) Abstract: The invention concerns a microcircuit card comprising at least one computer object and a register (Registry) comprising a logic identifier (myCalculator) of said object and at least one local reference (A000000000H, 0060H) of said object pertaining to the card. Said card further comprises means (CardManager) adapted to communicate, upon reception of a first message (lookup_APDU) including the logic identifier (myCalculator), at least one local code (K(0060H)) obtained from the local reference (0060H).

[Suite sur la page suivante]

WO 2004/042572 A2

FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : Cette carte à microcircuit comporte au moins un objet informatique et un registre (Registry) comportant d'une part un identifiant logique (myCalculator) de cet objet et au moins une référence (A000000000H, 0060H) de cet objet locale à la carte. Elle comporte également des moyens (CardManager) adaptés à communiquer, sur réception d'un premier message (lookup_ APDU) comportant l'identifiant logique (myCalculator), au moins un code local (K(0060H)) obtenu à partir de la référence locale (0060H).

Carte à microcircuit comportant des moyens de publication de ses objets informatiques

La présente invention concerne une carte à microcircuit du type de celles pouvant être insérées dans un système informatique, la carte comportant un système hôte de traitement de données (processeur, mémoires,...) et au moins un objet informatique (programme logicielle, variable, fichier,...) utilisé par ce système hôte.

Dans la suite de ce document on conviendra d'appeler :

- "terminal", le système informatique comportant un lecteur de carte à microcircuit dans lequel la carte est insérée. Le terminal peut par exemple être un ordinateur personnel (PC), un appareil de télécommunications (téléphone, assistant personnel,...) ou un système de traitement de données numériques conçu pour effectuer un traitement particulier, par exemple des traitements cryptographiques ;
- "système tiers", un système informatique relié au terminal tel que défini ci-dessus par un réseau de télécommunications (réseau local, réseau Internet ou autre).

De façon connue, la norme Java Card 2.2 définit un langage à objets permettant de simplifier le développement d'applications logicielles s'exécutant sur une carte à microcircuit. Le système hôte de la carte à microcircuit comporte traditionnellement un système d'exploitation (OS, "Operating System" en anglais) gravé dans une mémoire morte (ROM) de la carte, ce système d'exploitation comportant une machine virtuelle Java Card (JVM).

Selon cette norme, la communication entre le système hôte de la carte et le terminal s'effectue au moyen d'un protocole de communication standard rudimentaire.

Plus précisément, les informations échangées entre le système hôte et le terminal sont codées en hexadécimal et placées dans un tableau d'octets qui constitue soit un message de commande du terminal vers la carte, soit un

message de réponse de la carte vers le terminal. Un tel message de commande ou de réponse est appelé APDU ("Application Protocol Data Unit" en anglais).

Malheureusement, la norme Java Card 2.2 ne permet pas de publier la liste des objets informatiques accessibles sur la carte.

Cette contrainte rend très difficile le développement d'applications logicielles s'exécutant sur un système terminal ou un système tiers. En effet, le développement d'applications logicielles s'exécutant sur un tel système nécessite le codage fastidieux des APDU, et en particulier la connaissance des codes hexadécimaux pour adresser les objets de la carte. Un exemple d'application logicielle conforme au standard Java Card 2.2 est donné à titre d'exemple à l'annexe A.

La présente invention vise à résoudre les problèmes précités. Plus précisément, et selon un premier aspect, l'invention vise une carte à microcircuit comportant au moins un objet informatique, la carte étant caractérisée en ce qu'elle comporte :

- un registre comportant un identifiant logique de cet objet et au moins une première référence de cet objet locale à la carte; et
- des moyens adaptés à communiquer, sur réception d'un premier message comportant l'identifiant logique, au moins une deuxième référence locale obtenu à partir de la première référence locale.

Corrélativement, et selon un deuxième aspect, l'invention vise un équipement informatique de type terminal comportant des moyens adaptés à mettre en œuvre une application logicielle comportant au moins une première instruction pour utiliser au moins un objet informatique dans une carte à microcircuit, la première instruction utilisant un identifiant logique de cet objet. Le terminal est caractérisé en ce qu'il comporte des moyens d'obtention, à partir de l'identifiant logique, d'au moins une deuxième référence locale obtenu par la carte à microcircuit à partir d'une première référence de l'objet informatique locale à la carte ; des moyens de traduction de la première instruction en au moins une deuxième instruction pouvant être exécutée sur ladite carte, la deuxième instruction utilisant cette deuxième référence locale; et des moyens de communication adaptés à communiquer la deuxième instruction à la carte en vue de ladite utilisation.

Ainsi, une application logicielle s'exécutant sur le terminal peut comporter des instructions pour exécuter un objet informatique de la carte, ces instructions utilisant un identifiant logique de cet objet au lieu d'une référence locale à la carte.

D'autre part, cette application logicielle peut comporter des instructions de haut niveau pour l'échange d'informations entre le terminal et le système hôte de la carte, ces instructions de haut niveau étant traduites, par les moyens de traduction précités en messages APDU.

La présente invention facilite ainsi le développement d'applications logicielles s'exécutant sur le terminal, le développeur n'ayant ni à connaître les codes hexadécimaux pour adresser les objets de la carte à microcircuit, ni à coder les commandes APDU.

L'objet informatique de la carte à microcircuit peut notamment être un programme informatique, une variable ou un fichier informatique.

Selon une caractéristique avantageuse, la carte à microcircuit comporte en outre des moyens de publication de l'identifiant logique et de la première référence locale dans le registre de la carte.

Dans un premier mode de réalisation, le registre de la carte est créé une fois pour toutes, c'est à dire de façon statique au moment de la création de la carte à microcircuit.

En revanche, dans un mode préféré de réalisation conforme à la présente invention, les moyens de publication permettent la publication dynamique de l'identifiant logique et de la première référence locale de l'objet informatique dans le registre de la carte. Ainsi, seuls les objets informatiques nécessaires à un moment donné sont accessibles aux applications logicielles s'exécutant sur le terminal.

Dans un mode préféré de réalisation, l'objet informatique est un objet de type Java Card appartenant à une applet Java Card, et la deuxième référence locale de l'objet informatique est conforme à la norme Java Card. Dans ce mode préféré de réalisation, les moyens d'obtention de l'équipement informatique terminal sont adaptés à obtenir de la carte à microcircuit cette deuxième référence locale conforme à la norme Java Card.

Préférentiellement, la publication est effectuée à l'initialisation de cette applet.

Ce mode préféré de réalisation laisse au développeur de l'applet Java s'exécutant sur le système hôte de la carte à microcircuit, la possibilité de rendre chaque objet informatique de cette applet, accessible ou non aux applications logicielles s'exécutant sur le terminal.

Dans une variante de ce mode préféré de réalisation, les moyens de communication sont adaptés à communiquer un identifiant de l'applet sur réception du premier message.

On obtient ainsi dans cette variante l'identifiant de l'applet sans que le programmeur n'ait besoin de connaître son code hexadécimal. Conformément à la norme Java Card, cet identifiant est nécessaire pour pouvoir utiliser ultérieurement, les objets informatiques de cette applet.

Dans une autre variante préférée de réalisation de la carte à microcircuit selon l'invention, les moyens de communication de la carte communiquent, sur réception d'un deuxième message, l'ensemble des identifiants logiques compris dans le registre de la carte.

Cette variante est particulièrement avantageuse car elle permet au système terminal de connaître tous les objets informatiques publiés par la carte à microcircuit et les différentes versions de ces objets.

Dans une première variante de réalisation, la deuxième référence locale communiqué par les moyens de communication de la carte à microcircuit est la première référence locale de l'objet informatique proprement dite.

Dans une variante préférée de réalisation, cette deuxième référence locale est temporaire et est obtenue par cryptage de la première référence locale en utilisant une clef de cryptage de la carte à microcircuit.

Ainsi, la deuxième référence locale ne peut être utilisée que pour une session définie par la sélection de l'applet. Cette variante préférée de réalisation permet d'augmenter la sécurité de la carte, conformément aux recommandations de la norme Java Card 2.2, afin de se prémunir de attaques par "rejeux", c'est à dire des attaques visant à utiliser frauduleusement l'objet de la carte, en réutilisant sa référence, sans sélectionner l'applet implémentant cet objet.

Dans un mode préféré de réalisation, le terminal comporte en outre des moyens de publication dans un de ses registres d'un objet tampon comportant une interface identique à celle de l'objet informatique de la carte, cet objet tampon étant adapté à traduire une instruction s'exécutant sur un système tiers et utilisant l'identifiant logique, en au moins une deuxième instruction pouvant être exécutée sur la carte et utilisant la deuxième référence locale.

Ce mode préféré de réalisation permet l'utilisation des objets informatiques de la carte à microcircuit par une application logicielle s'exécutant sur un système tiers en réseau avec le terminal de la carte.

Préférentiellement, ce registre est conforme au standard "Java2 SE RMI Registry".

Dans une variante préférée de ce mode de réalisation, les moyens de publication obtiennent et publient dans ce registre du système terminal l'ensemble des objets tampons des objets informatiques publiés par ladite carte.

Cette variante est particulièrement avantageuse car elle permet au système tiers de connaître tous les objets informatiques publiés par la carte à microcircuit et les différentes versions de ces objets.

L'invention sera mieux comprise et d'autres avantages apparaîtront plus clairement à la lumière de la description qui va suivre d'une carte à microcircuit et d'un système informatique terminal conformes à son principe, cette description étant donnée uniquement à titre d'exemple et faite en référence aux dessins annexés dans lesquels :

- la figure 1 est un schéma bloc d'une carte à microcircuit conforme à la présente invention, dans un mode particulier de réalisation ; et
- la figure 2 est un schéma bloc d'un système informatique terminal conforme à l'invention dans un mode préféré de réalisation.

Par ailleurs la description est accompagnée des annexes suivantes :

- annexe A : code source d'une applet Java Card de la carte à microcircuit de la figure 1 ;
- annexe B : première partie du code source d'un code client du système informatique terminal de la figure 2;
- annexe C : tableau de syntaxe d'une commande APDU pour obtenir l'identificateur d'une applet;

- annexe D : tableau de syntaxe d'une commande APDU pour obtenir la référence d'un objet;
- annexe E : tableau de syntaxe d'une commande APDU d'affectation d'une valeur à un objet informatique conformément à la norme ISO7816 ;
- annexe F : deuxième partie du code source d'un code client du système informatique terminal de la figure 2;
- annexe G : tableau de syntaxe d'une commande APDU pour exécuter un programme à distance sur une carte à microcircuit; et
- annexe H : : tableau de syntaxe d'une commande APDU pour obtenir les identifiants logiques de tous les objets d'une carte à microcircuit.

La **figure 1** représente une carte à microcircuit 10 conforme à la présente invention. Cette carte 10 comporte un système hôte de traitement de données comportant un processeur CPU, associé de façon classique à une mémoire morte ROM, une mémoire volatile RAM, et une mémoire non volatile réinscriptible EEPROM.

La mémoire morte ROM comporte un système d'exploitation OS, ce système d'exploitation OS comportant une machine virtuelle Java Card JVM.

La carte à microcircuit 10 comporte des moyens d'entrée-sortie E/S adaptés à émettre des données numériques à destination d'un terminal et à recevoir des données numériques en provenance de ce terminal.

Ces moyens d'entrée-sortie E/S sont par exemple constitués par un port série connu de l'homme du métier.

La carte à microcircuit 10 décrite ici comporte trois objets informatiques.

De façon connue de l'homme du métier, un objet conforme à la norme Java Card est un cas particulier de programme informatique. Par souci de simplification, on conviendra dans la suite de ce document qu'un objet peut être vu comme un programme informatique.

La carte à microcircuit 10 comporte ainsi un premier objet informatique, à savoir un programme informatique appartenant à une applet Java Card CalculatorApplet dont le code, donné à l'annexe A, est mémorisé dans une mémoire réinscriptible EEPPROM.

Cet objet informatique comporte une première référence 0060H locale à la carte 10 conforme à la norme Java Card. Selon l'art antérieur, le programmeur d'une application logicielle s'exécutant sur le terminal doit utiliser cette référence 0060H pour exécuter ce programme sur la carte.

Conformément à la présente invention, ce programme informatique comporte également un identifiant logique "myCalculator" pouvant être utilisé par un programmeur du système informatique terminal à la place de la référence 0060H.

La carte à microcircuit 10 comporte un deuxième objet informatique, à savoir un fichier mémorisé dans la mémoire non volatile réinscriptible EEPROM.

On notera tout d'abord, que le mot "fichier" doit, dans ce document, être compris au sens large, ce mot désignant aussi bien un fichier qu'un répertoire. Notamment, le mot fichier désignera en particulier un répertoire DF (Dedicated file) et un fichier EF (Elementary File) au sens de la norme ISO7816.

De façon connue de l'homme du métier des cartes à microcircuit, le fichier comporte, conformément à la norme ISO7816, une première référence locale à la carte appelée FID (File Identifier) codée sur deux octets en hexadécimal, soit dans l'exemple décrit ici "3F00H".

Conformément à la présente invention, ce fichier comporte un identifiant logique "CARD_HOLDER".

La carte à microcircuit 10 comporte un troisième objet informatique, à savoir une variable (ou "data objet") dans un registre de la mémoire réinscriptible EEPROM.

Cette variable comporte une première référence locale à la carte codée sur deux octets en hexadécimal, soit dans l'exemple décrit ici "0050H". On supposera que cette variable sert à mémoriser une date.

Conformément à la présente invention, cette variable comporte un identifiant logique, ici la chaîne de caractères "date".

Conformément à la présente invention, la carte à microcircuit comporte un fichier 20 mémorisant une table comportant une ligne pour chaque objet informatique accessible par une application logicielle s'exécutant sur un système informatique terminal ou sur un système informatique tiers.

Chacune des lignes de la table du fichier 20 comporte :

- un enregistrement comportant l'identifiant logique de l'objet informatique ; et
- un enregistrement comportant une première référence de cet objet locale à la carte.

La carte à microcircuit 10 comporte une application CardManager mémorisée par exemple en mémoire ROM.

De façon connue, l'application CardManager est adaptée, en utilisant les moyens d'entrée-sortie E/S, à échanger des messages constitués de commandes et de réponses APDU avec le système informatique terminal dans lequel la carte à microcircuit est insérée.

Elle est en particulier adaptée à recevoir un message Lookup_APDU en provenance du terminal précité, ce message comportant l'identifiant logique d'un objet informatique.

Cette application CardManager est également adaptée à communiquer au terminal, en utilisant le fichier 20, une deuxième référence locale de l'objet informatique, sur réception du message Lookup_APDU.

Plus précisément, l'application CardManager extrait l'identifiant logique contenu dans le message APDU, recherche dans le fichier 20 la ligne comportant cet identifiant logique dans sa deuxième colonne et obtient la première référence locale contenue dans la deuxième colonne de cette même ligne.

Dans un mode de réalisation décrit ici, la première référence locale ainsi obtenue est ensuite cryptée avec une fonction K qui utilise une clef de cryptage KEY mémorisée en mémoire réinscriptible EEPROM, ce qui permet ainsi d'obtenir une deuxième référence locale communiquée au terminal par l'application CardManager.

En variante, la deuxième référence locale communiquée au terminal est la première référence locale elle-même.

Dans le mode de réalisation préféré, les moyens de communication sont adaptés à communiquer, sur réception d'un message comportant l'identifiant logique d'un objet informatique conforme à la norme Java Card 2.2, l'identifiant AID de l'applet Java Card CalculatorApplet comportant cet objet informatique, soit par exemple la référence A000000000H.

Ainsi, sur réception d'un message comportant l'identifiant logique "myCalculator", les moyens de communication communiquent d'une part l'identificateur AID A000000000H de l'applet Java CalculatorApplet et d'autre part la deuxième référence locale K(0060H)= 0460h obtenue par cryptage, avec la clef KEY, de la référence locale 0060H.

L'application CardManager fournit également le nom Calculator de l'interface de l'objet informatique.

Par ailleurs :

- sur réception d'un message comportant l'identifiant logique "CARD_HOLDER", les moyens de communication communiquent la deuxième référence locale K(0004H) ; et
- sur réception d'un message comportant l'identifiant logique "date", les moyens de communication communiquent la deuxième référence locale K(0050H).

Dans un premier mode de réalisation, le fichier 20 est construit de façon statique, au moment de la création de la carte. Il s'agit par exemple d'un tableau mémorisé dans la mémoire morte ROM.

Dans le mode préféré de réalisation décrit ici, le fichier 20 est mis à jour de façon dynamique. A cette fin, la carte à microcircuit 10 comporte des moyens de publication de l'identifiant logique et de la première référence locale dans le fichier 20 de la carte.

Ces moyens de publication sont constitués ici par un programme informatique Binding_API mémorisé en mémoire ROM, ce programme informatique Binding_API comportant une instruction informatique bind, dont un exemple d'utilisation pour la publication du programme "myCalculator" est donné à la ligne A33 de l'annexe A.

Préférentiellement et tel que décrit ici en référence à l'annexe A, cette publication s'effectue à l'initialisation de l'applet Java CalculatorApplet.

Nous allons maintenant décrire en référence à la **figure 2**, un équipement informatique 100 de type terminal conforme à la présente invention.

Dans l'exemple décrit ici, le terminal 100 est constitué par un ordinateur personnel PC, comportant des moyens 110 d'insertion d'une carte à microcircuit 10 telle que décrite précédemment en référence à la figure 1.

Ce terminal 100 comporte notamment des moyens connus pour la mise en œuvre d'une application logicielle, à savoir un processeur CPU associé à des mémoires ROM et RAM et un disque dur DISK.

Le terminal 100 comporte également des moyens d'entrée-sortie E/S, adaptés à émettre des données numériques à destination d'une carte à microcircuit conforme à l'invention, et à recevoir des données d'une telle carte.

Ces moyens d'entrée-sortie E/S sont par exemple constitués par un port série connu de l'homme du métier.

Conformément à la présente invention, le terminal 100 peut mettre en œuvre une application logicielle DesktopApp mémorisée dans le disque dur DISK comportant au moins une première instruction pour utiliser un objet informatique dans une carte à microcircuit, cette première instruction utilisant l'identifiant logique de cet objet.

Le code de l'application logicielle DesktopApp est donné aux lignes B21 à B31 de **l'annexe B**. Ce code comporte en particulier une ligne B23 utilisant l'identifiant logique "myCalculator" du programme informatique définie précédemment en référence à la figure 1.

Afin de pouvoir utiliser cet identifiant logique, et pour assurer une compatibilité avec la norme Java Card 2.2, l'équipement informatique terminal 100 comporte des moyens d'obtention, à partir de cet identifiant logique, en coopération avec les moyens d'entrée-sortie E/S, d'une deuxième référence locale. Cette deuxième référence locale est obtenue par la carte à microcircuit 10 à partir d'une première référence de l'objet informatique locale à la carte à microcircuit.

Dans le mode de réalisation décrit ici, ces moyens d'obtention sont réalisés par un programme CardNaming.lookup mémorisé sur le disque dur DISK, et dont le code source est donné aux lignes B12 à B20 de l'annexe B.

Dans le mode de réalisation décrit ici, les moyens d'obtention obtiennent tout d'abord l'identifiant AID de l'applet CalculatorApplet comportant l'objet informatique :

(B14) :byte[ ]AID= CardNaming.cardManagerLookup("myCalculator");

Dans l'exemple décrit ici, la méthode CardNaming.cardManagerLookup construit une première commande APDU contenant l'identifiant logique myCalculator et envoie cette commande APDU à la carte.

La carte retourne en réponse l'identifiant AID correspondant, soit la valeur A000000000H.

Cette première commande APDU provoque l'exécution sur la carte de l'instruction Lookup_AID de l'application CardManager qui obtient, à partir du fichier 20, la valeur de l'AID associée à l'identifiant logique myCalculator.

De façon connue, la première commande APDU est envoyée (après l'envoi d'une commande APDU de sélection de l'application CardManager, conformément à la norme ISO7816) par la méthode CardNaming.cardManagerLookup en utilisant la classe CardService prédéfinie par le consortium OCF (Open Card Framework) permettant d'envoyer des APDU à une carte. La syntaxe de cette commande APDU est donnée à l'**annexe C**.

Dans un deuxième temps, le programme CardNaming.appletLookup envoie une deuxième commande APDU avec deux paramètres d'entrée, à savoir l'identifiant AID (A000000000H) de l'applet obtenu précédemment et l'identifiant logique myCalculator de l'objet informatique de la carte.

Le programme CardNaming.appletLookup obtient en retour la deuxième référence locale temporaire K(0060H)=0460h obtenue par cryptage à partir de la première référence locale 0060H de cet objet sur la carte à microcircuit.

(B15) byte[]ref= CardNaming.appletLookup(AID, "myCalculator");

Cette deuxième commande APDU, dont la syntaxe est donnée à l'**annexe D**, provoque l'exécution sur la carte de l'instruction lookup_reference de l'application CardManager qui obtient, à partir du fichier 20, la première référence locale 0060H associée à l'identifiant logique myCalculator, puis calcule la deuxième référence locale 0460h.

L'équipement informatique terminal 100 comporte en outre des moyens de traduction de la première instruction en au moins une deuxième instruction pouvant être exécutée sur ladite carte, cette deuxième instruction utilisant ladite au moins une deuxième référence locale.

Ainsi par exemple, lorsque l'application logicielle DesktopApp désire modifier le contenu de la variable de la carte à microcircuit 10 dont l'identifiant

logique est "date" avec la valeur "01/01/2002", les moyens de traduction génèrent une deuxième instruction, sous la forme d'une commande Put_data APDU, dont la syntaxe est donnée à l'**annexe E**. Cette instruction utilise la première référence locale 0050H de cet objet.

Dans le mode de réalisation décrit ici, les moyens de traduction sont constitués par un objet proxy créé à partir de la deuxième référence locale de l'objet obtenue précédemment (0460H). Dans ce mode de réalisation, l'objet proxy est une instance de la classe Proxy connue de l'homme du métier, et définie par Java 2™ platform, Standard Edition V1.3.

Conformément au standard Java 2™ platform, Standard Edition V1.3, on construit une classe CardHandler qui comporte une méthode invoke permettant la traduction des appels de méthode en coopération avec l'objet proxy. La méthode invoke utilise la classe CardService définie par le consortium OCF (Open Card Framework). Le pseudo code de la classe CardHandler est donné aux lignes B1 à B11 de l'annexe B.

Dans l'exemple décrit ici, l'objet proxy est construit (lignes B16 et B17) et retourné (ligne B18) par la méthode CardNaming.lookup.

Ainsi, après l'exécution de l'instruction de la ligne B23 de l'annexe B, un objet calculator est créé dans le terminal, cet objet permettant la traduction des instructions utilisant l'objet informatique de la carte dont la référence logique est « myCalculator ».

La deuxième référence ref, locale à la carte, de cet objet calculator de traduction est mémorisée par le constructeur de la classe CardHandler décrite ci-dessus (ligne B4, annexe B).

L'objet de traduction calculator peut ensuite être utilisé dans un programme du terminal pour exécuter des méthodes de l'objet de la carte à microcircuit dont la référence logique est « myCalculator ».

Le développeur d'un tel programme peut par exemple écrire la ligne B24 de l'annexe B pour exécuter, sur la carte à microcircuit, l'addition des nombres 5 et 6.

A l'exécution de l'instruction de la ligne B24, la méthode Invoke de l'objet CardHandler décrite précédemment, créé et envoie la commande APDU Invoke

conformément à la norme Java 2.2. La syntaxe de cette commande est donnée à l'**annexe G**.

Préalablement, l'applet CalculatorApplet a été sélectionnée au moment de l'exécution de la ligne B15.

Conformément à la norme Java Card 2.2, l'instruction invoke ainsi communiquée à l'applet CalculatorApplet au moyen de cette commande APDU déclenche l'exécution de la méthode add avec les paramètres 5 et 6.

La présente invention permet en outre, à une application logicielle s'exécutant sur un système tiers d'accéder à un objet informatique de la carte à microcircuit en utilisant son identifiant logique, lorsque la carte est insérée dans un terminal relié à ce système tiers par un réseau de télécommunications.

A cette fin, et dans un mode préféré de réalisation, l'équipement informatique terminal 100 comporte des moyens de publication dans un registre "Standard RMI Registry" d'une mémoire volatile RAM de ce terminal, d'un objet tampon remoteCalculator, présentant une interface identique à celle de l'objet informatique de la carte à microcircuit, et permettant à une application informatique s'exécutant sur un système tiers, d'utiliser cet objet informatique.

Pour cela, un objet intermédiaire calculator_invoker est tout d'abord créé (ligne B25, annexe B) à partir de l'objet de traduction calculator décrit précédemment.

L'objet calculator_invoker appartient à la classe InvokerImpl, cette classe étant elle-même une classe UnicastRemoteObject connue de l'homme du métier et implémentant les mécanismes RMI de Java SE 1.3. Le code de la classe InvokerImpl est donné à l'annexe F, lignes F3 à F8.

De façon connue, l'objet intermédaire calculator_invoker comporte l'objet de traduction calculator et une méthode invokeMethod, cette méthode invokeMethod implémentant un mécanisme de traduction des appels aux méthodes de l'objet calculator, en utilisant le mécanisme de réflectivité du langage Java (ligne F7).

L'objet calculator_invoker créé à la ligne B25 est un objet RMI mémorisé dans le registre du terminal "Standard RMI Registry", mais il ne possède pas les méthodes des objets informatiques présents sur la carte à microcircuit. En particulier, il ne possède pas la méthode add.

Pour permettre l'utilisation de cette méthode par une application s'exécutant sur un système tiers, on mémorise dans le registre du terminal "Standard RMI Registry" l'objet tampon remoteCalculator (lignes B27 à B29) qui implémente les méthodes de l'objet de la carte dont l'identifiant logique est "myCalculator", cet objet remoteCalculator étant identifié également par l'identifiant logique "myCalculator" (ligne B29). L'objet remoteCalculator contient en outre l'identifiant logique et les moyens d'appel RMI à l'objet intermédiare calculator_invoker, comme on peut le voir dans le code de la classe InvokerHandler ligne F9 à F15. L'objet tampon remoteCalculator est adapté, en faisant appel à l'objet calculator_invoker à traduire une instruction s'exécutant sur le système tiers et utilisant ledit identifiant logique, en au moins une deuxième instruction pouvant être exécutée sur ladite carte et utilisant la deuxième référence locale (0460H).

Deux objets sont ainsi enregistrés dans le registre du terminal "Standard RMI registry", à savoir d'une part l'objet intermédiaire calculator_invoker sous l'identifiant "myCalculator_Invoker" (ligne B26) et l'objet tampon remoteCalculator avec l'identifiant logique "myCalculator" (ligne B29).

Ainsi, lorsque une application logicielle du système tiers exécute l'instruction suivante :

Calculator calculator_tiers = (Calculator)Naming.lookup("myCalculator"),

la méthode Naming.lookup de Java2 SE lit, dans le registre "Standard RMI registry" du système terminal, l'objet tampon remoteCalculator, le reconstruit dans le système tiers, et l'affecte à l'objet calculator_tiers.

L'application logicielle du système tiers peut ensuite utiliser l'objet calculator_tiers pour accéder à l'objet correspondant de la carte à microcircuit 10 via l'objet intermédiaire calculator_invoker du système terminal.

Ainsi, les objets informatiques de la carte à microcircuit 10 peuvent être utilisés sur un système informatique tiers connecté en réseau au système informatique terminal 100.

Dans le mode préféré de réalisation décrit ici, le système terminal fournit également les moyens de publication décrits précédemment pour la publication de l'objet calculator sous forme d'une classe Java BindingService dont les

principales instructions, nécessaires à la compréhension, sont données à l'annexe F, lignes F16 à F30.

Ces moyens de publication BindingService sont en particulier adaptés à obtenir et publier dans le registre "standard RMI Registry" du terminal l'ensemble des objets tampons des objets informatiques publiés par la carte à microcircuit.

Dans le mode préféré de réalisation décrit ici, les moyens de publication BindingService utilisent à cette fin la commande APDU get_bound_objects_APDU dont la syntaxe est donnée à l'annexe G.

ANNEXES

ANNEXE A

```
A1    public interface Calculator extends Remote {
A2        short add (short a, short b) throws RemoteException ;
A3    }

A10   public class CalculatorImpl extends CardRemoteObject implements Calculator {
A11       public CalculatorImpl ( ) { super ( ); }
A12       short add(short a, short b) throws RemoteException {
A13           return (short) (a+b) ;
A14       }
A15   }

A20   public class CalculatorApplet extends Applet {
A21       private Dispatcher dispatcher ;
A22       private final static byte CALCULATOR = { (byte)m, (byte)y, (byte)C, (byte)a, (byte)l
A23           , (byte)c, (byte)u, (byte)l, (byte)a, (byte)t, (byte)o, (byte)r };

A30       private CalculatorApplet{
A31       Calculator calculator = new CalculatorImpl ( );
A32       OCSRMIService rmi = new OCSRMIService (calculator) ;
A33       RMIRegistry.bind   (CALCULATOR, (short) 0, (byte) CALCULATOR.length,
A34                           calculator) ;
A35       dispatcher = new Dispatcher(1);
A36       dispatcher.addService(rmi, Dispatcher.PROCESS_COMMAND);
A37       }

A40       public static void install(byte[ ] buffer, short offset, byte length) {
A41           (new CalculatorApplet( )).register( );
A42       } ;

A50       public void process (APDU apdu) {install(byte[ ] buffer, short offset, byte length) {
A51       dispatcher.process (apdu);
A52       }
A53   }
```

ANNEXE B

```
B1    class CardHandler implements Invocation Handler {
B2        private byte [ ] ref ;

B3        CardHandler(byte[ ] ref) {
B4            this.ref = ref ;
B5        }

B6        public Object invoke(Objet proxy, Method method, Object[ ] params) {
B7            /* Code de traduction de l'appel à la méthode method */
B8            /* Construction d'une APDU, en utilisant ref pour lancer l'exécution
              de la méthode method sur la carte */
B9            /* Envoi de l'APDU à la carte et récupération de la réponse retournée
              par la méthode */
B10       }
B11   }

B12   class CardNaming {
B13       public static Remote lookup(String name) {
B14           byte aid[] = CardNaming.cardManagerLookup (name) ;
B15           byte ref[] = CardNaming.appletLookup(aid, name) ;
B16           Cardhandler ch = new Cardhandler(K(ref)) ;
B17           Remote proxy = (Remote) Proxy.newProxyInstance(..., ch, ...) ;
B18           return proxy ;
B19       }
B20   }

B21   public class DesktopApp {
B22       void main (String [ ] args) {
B23           Calculator calculator = (Calculator) CardNaming.lookup ("myCalculator");
B24           short result = calculator.add((short) 5, (short) 6);
B25           InvokerImpl calculator_invoker = new InvokerImpl(calculator);
B26           Naming.bind("myCalculator_invoker",calculator_invoker);
B27           InvokerHandler handler = new InvokerHandler("myCalculator_invoker");
B28           Remote remoteCalculator = Proxy.newProxyInstance(..., handler,...);
B29           Naming.bind("myCalculator", remoteCalculator);
B30       }
B31   }
```

ANNEXE C

AID Lookup_APDU		
Nom du champs (ISO 7816)	Valeur en hexadécimal	Description du contenu du champs
CLA	80h	Classe d'appel propriétaire
INS	42h	Instruction lookup_AID
P1	00h	Pas de paramètre
P2	00h	Pas de paramètre
Lc	0Ch	Longueur de l'identifiant logique myCalculator
Data Field	6D7943616C63756C61746F72h	Identifiant logique myCalculator codé en format UTF-8
Le	00h	Taille attendue de la réponse inconnue

ANNEXE D

Référence Lookup_APDU		
Nom du champs (ISO 7816)	Valeur en hexadécimal	Description du contenu du champs
CLA	80h	Classe d'appel propriétaire
INS	40h	Instruction lookup_référence
P1	00h	Pas de paramètre
P2	10h	Paramètre spécial
Lc	0Ch	Longueur de l'identifiant logique
Data Field	6D7943616C63756C61746F72h	Identifiant logique codé en format UTF-8
Le	00h	Taille attendue de la réponse inconnue

ANNEXE E

Put_data APDU		
Nom du champs (ISO 7816)	Valeur en hexadécimal	Description du contenu du champs
CLA	80h	Classe d'appel propriétaire
INS	Dah	Instruction Put_data
P1	00h	premier octet du tag date
P2	50h	deuxième octet du tag date
Lc	04h	Longueur de la date 01/01/2002
Data	01012002h	Date 01/01/2002

ANNEXE F

```
F1    public interface Invoker extends Remote {
F2        public Object invoke(OMethod method, Object[] args); }

F3    public class InvokerImpl extends UnicastRemoteObject implements Invoker {
F4        private Remote cardObject;
F5        InvokerImpl(Remote cardObject) {Super(); this.cardObject = cardObject;}
F6        public Object invokeMethod(String method, Object[] args) {
F7                /** use Reflectivity **/        }
F8    }

F9    public class InvokerHandler implements InvocationHandler, Serializable {
F10       private String invokerName;
F11       InvokerHandler(String invokerName) {this.invokerName = invokerName;}
F12       public Object invoke(Object proxy, Method method, Object[] args) {
F13           Invoker invoker = Naming.lookup(invokerName);
F14           return invoker.invokeMethod( getString(method), args); }
F15   }

F16   public class BindingService {
F17       public static main(String[] args) {
F18           String registryURL = args[1];
F19           Iterator iterator = getBoundObjects();
F20           while (iterator.hasNext()) {
F21               String name = (String)iterator.next();
F22               Remote cardObject = CardNaming.lookup(name);
F23               InvokerImpl invoker = new InvokerImpl(cardObject);
F24               String invokerName = registryURL + name + "_Invoker";
F25               Naming.bind(invokerName, invoker);
F26               InvokerHandler handler = new InvokerHandler(invokerName);
F27               Remote remoteObject = Proxy.nexProxyInstance(..., handler,...);
F28               Naming.bind(registryURL + name, remoteObject);
F29           }
F30   }
F31 }
```

ANNEXE G

Invoke APDU		
Nom du champs (ISO 7816)	Valeur en hexadécimal	Description du contenu du champs
CLA	80h	Classe d'appel propriétaire
INS	38h	Instruction Invoke
P1	02h	premier octet de version Java Card
P2	02h	deuxième octet de version Java Card
Lc	08h	Longueur des données
Data	0460 A7F6 0005 0006h	Données :référence (2 octets), méthode add (2 octets), paramètre "5" (2 octets) et paramètre "6" (2 octets).

ANNEXE H

get_bound_objects APDU		
Nom du champs (ISO 7816)	Valeur en hexadécimal	Description du contenu du champs
CLA	80h	Classe d'appel propriétaire
INS	44h	Instruction get_bound_objects
P1	00h	Pas de paramètre
P2	00h	Pas de paramètre
Le	00h	Taille attendue de la réponse

REVENDICATIONS

1. Carte à microcircuit comportant au moins un objet informatique caractérisée en ce qu'elle comporte :

- un registre (20) comportant un identifiant logique (myCalculator) dudit objet et au moins une première référence (0060H) dudit objet locale à ladite carte; et
- des moyens (CardManager) adaptés à communiquer, sur réception d'un premier message (lookup_APDU) comportant ledit identifiant logique (myCalculator), au moins une deuxième référence locale (K(0060H)= 0460H) obtenue à partir de ladite au moins une première référence locale (0060H).

2. Carte à microcircuit selon la revendication 1, caractérisée en ce qu'elle comporte en outre des moyens de publication (bind) dudit identifiant logique (myCalculator) et de ladite au moins une première référence locale (0060H) dans ledit registre (20) de la carte.

3. Carte à microcircuit selon la revendication 1 ou 2 dans laquelle ledit objet informatique est un objet de type Java Card appartenant à une applet Java Card (CalculatorApplet), la carte étant caractérisée en ce que ladite deuxième référence locale (0460H) dudit objet informatique est conforme à la norme Java Card.

4. Carte à microcircuit selon les revendications 2 et 3, caractérisée en ce que ladite publication est effectuée à l'initialisation de ladite applet (CalculatorApplet).

5. Carte à microcircuit selon la revendication 3 ou 4, caractérisée en ce que les moyens de communication (card manager) sont adaptés à communiquer un identifiant (A000000000H) de ladite applet sur réception dudit premier message (lookup_APDU).

6. Carte à microcircuit selon l'une quelconque des revendications 1 à 5, caractérisée en ce que ledit objet informatique est un programme informatique (myCalculator), une variable (date) ou un fichier informatique (CARD_HOLDER).

7. Carte à microcircuit selon l'une quelconque des revendications 1 à 6, caractérisée en ce que lesdits moyens de communication communiquent, sur réception d'un deuxième message (get_bound_objects_APDU) l'ensemble des identifiants logiques compris dans ledit registre (20).

8. Carte à microcircuit selon l'une quelconque des revendications 1 à 7, caractérisée en ce que ladite deuxième référence locale (A000000000H) est ladite première référence locale (A000000000H).

9. Carte à microcircuit selon l'une quelconque des revendications 1 à 7, caractérisée en ce que ladite deuxième référence locale (0460H) est temporaire et est obtenue par cryptage de la première référence locale (0060H) en utilisant une clef (KEY) de cryptage de la carte à microcircuit.

10. Equipement informatique (terminal) de type terminal comportant des moyens (CPU, ROM, RAM) adaptés à mettre en œuvre une application logicielle (DesktopApp) comportant au moins une première instruction pour utiliser au moins un objet informatique dans une carte à microcircuit, caractérisé en ce que ladite au moins une première instruction utilisant un identifiant logique (myCalculator) dudit objet, ledit équipement informatique comporte :

- des moyens d'obtention (CardNaming.lookup), à partir dudit identifiant logique (myCalculator), d'au moins deuxième référence locale (K(0060H) = 0460H) obtenue par la carte à microcircuit à partir d'une première référence (0060H) dudit objet informatique locale à ladite carte;
- des moyens de traduction (proxy, invoke) de ladite au moins une première instruction en au moins une deuxième instruction pouvant être exécutée sur ladite carte, ladite au moins une deuxième instruction utilisant ladite au moins deuxième référence locale (0460H); et

- des moyens de communication adaptés à communiquer ladite au moins une deuxième instruction à ladite carte en vue de ladite utilisation.

11. Equipement informatique selon la revendication 10 dans lequel ledit objet informatique est un objet de type Java Card appartenant à une applet Java Card (CalculatorApplet) de la carte à microcircuit, l'équipement informatique étant caractérisé en ce que les moyens d'obtention (CardNaming.lookup) sont adaptés à obtenir une deuxième référence (0460H) conforme à la norme Java Card, obtenu, par ladite carte, à partir d'une première référence (0060H) dudit objet informatique.

12. Equipement informatique selon la revendication 10, caractérisé en ce que les moyens d'obtention (CardNamingAPI lookup) sont adaptés à obtenir un identifiant (A000000000H) de ladite applet (CalculatorApplet).

13. Equipement informatique selon l'une quelconque des revendications 10 à 12, caractérisé en ce que ledit objet informatique est un programme informatique (myCalculator), une variable (date) ou un fichier informatique (CARD_HOLDER).

14. Equipement informatique selon l'une quelconque des revendications 10 à 13, caractérisé en ce qu'il comporte en outre des moyens de publication (BindingService), dans un registre dudit système informatique terminal (standard RMI Registry), d'un objet tampon (remoteCalculator) comportant une interface identique à celle de l'objet informatique de la carte, cet objet tampon étant adapté à traduire (invokeMethod) une instruction s'exécutant sur un système tiers et utilisant ledit identifiant logique, en au moins une deuxième instruction pouvant être exécutée sur ladite carte et utilisant ladite deuxième référence locale (0460H).

15. Equipement informatique selon la revendication 14, caractérisé en ce que les moyens de publication (BindingService) sont adaptés à obtenir et publier dans le registre dudit système informatique terminal (standard RMI

Registry), l'ensemble des objets tampons des objets informatiques publiés par ladite carte.

16. Equipement informatique selon la revendication 14 ou 15, caractérisé en ce que ledit objet informatique étant un objet de type Java Card, ledit registre (Java2 SE RMI Registry) est conforme au standard "Java standard RMI registry".

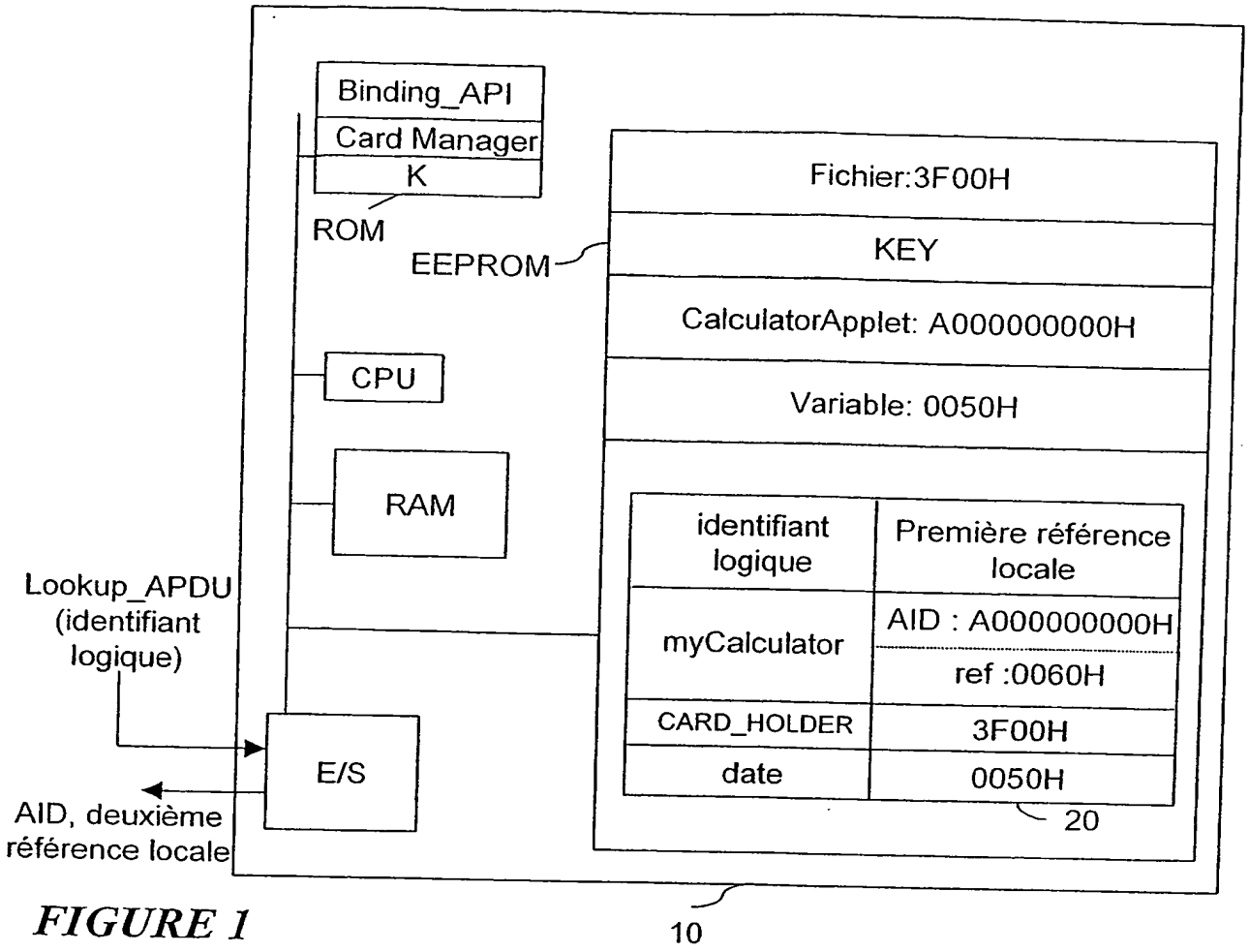


FIGURE 1

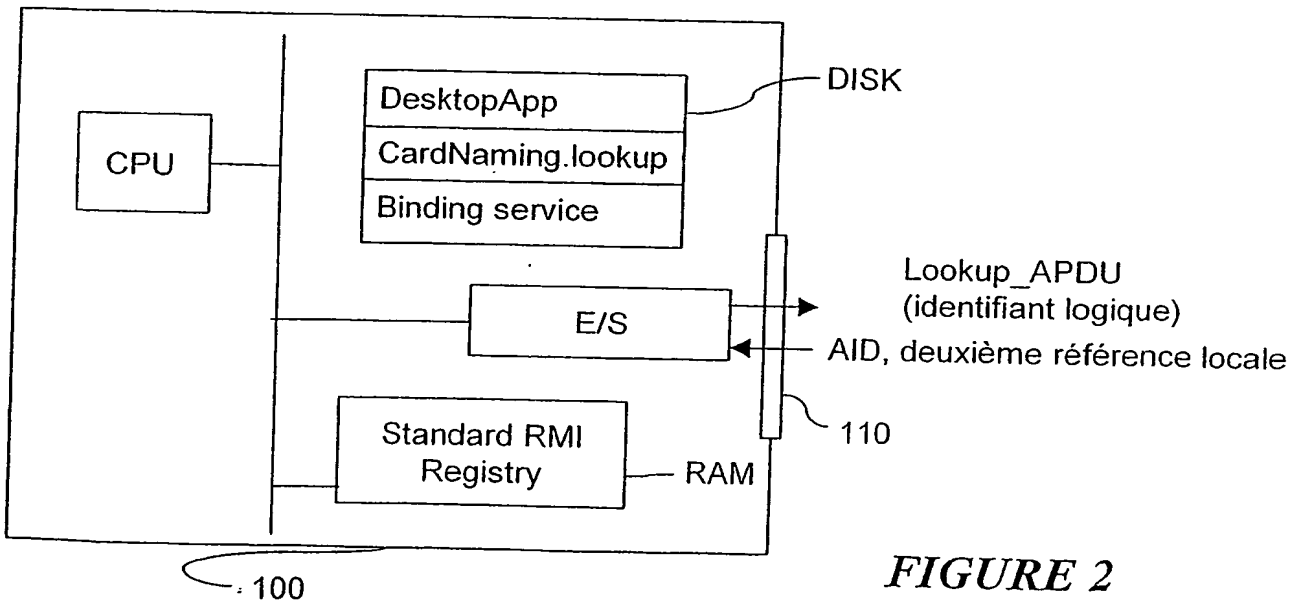


FIGURE 2